fc	fractional capability variable
x, g, a, b	expression variable
k	integer variable
el	array-element variable
l	location variable
M	matrix variable

m	::=			matrix expressions
	|	M_{k_1,k_2}		matrix variables
	|	$m + m'$		matrix addition
	|	$m\, m'$		matrix multiplication
f	::=			fractional capability
	|	fc		variable
	|	1		whole capability
	|	$\frac{1}{2} \cdot f$		
t	::=			linear type
	|	**unit**		unit
	|	**bool**		boolean (true/false)
	|	**int**		63-bit integers
	|	**elt**		array element
	|	f **arr**		arrays
	|	f **mat**		matrices
	|	$!t$		multiple-use type
	|	$\forall\, fc.t$	bind fc in t	frac. cap. generalisation
	|	$t \otimes t'$		pair
	|	$t \multimap t'$		linear function
	|	(t)	S	parentheses
p	::=			primitive
	|	**not**		boolean negation
	|	$(+)$		integer addition
	|	$(-)$		integer subtraction
	|	$(*)$		integer multiplication
	|	$(/)$		integer division
	|	$(=)$		integer equality
	|	$(\langle)$		integer less-than
	|	$(+.)$		element addition
	|	$(-.)$		element subtraction
	|	$(*.)$		element multiplication
	|	$(/.)$		element division
	|	$(= .)$		element equality
	|	$(< .)$		element less-than
	|	**set**		array index assignment
	|	**get**		array indexing
	|	**share**		share array
	|	**unshare**		unshare array
	|	**free**		free arrary
	|	**array**		Owl: make array
	|	**copy**		Owl: copy array
	|	**sin**		Owl: map sine over array
	|	**hypot**		Owl: $x_i := \sqrt{x_i^2 + y_i^2}$
	|	**asum**		BLAS: $\sum_i \lvert x_i \rvert$

	axpy		BLAS: $x := \alpha x + y$
	dot		BLAS: $x \cdot y$
	rotmg		BLAS: see its docs
	scal		BLAS: $x := \alpha x$
	amax		BLAS: $\operatorname{argmax} i : x_i$
	setM		matrix index assignment
	getM		matrix indexing
	shareM		share matrix
	unshareM		unshare matrix
	freeM		free matrix
	matrix		Owl: make matrix
	copyM		Owl: copy matrix
	copyM_to		Owl: copy matrix onto another
	sizeM		dimension of matrix
	trnsp		transpose matrix
	gemm		BLAS: $C := \alpha A^{T?} B^{T?} + \beta C$
	symm		BLAS: $C := \alpha AB + \beta C$
	posv		BLAS: Cholesky decomp. and solve
	potrs		BLAS: solve with given Cholesky

v	::=			values
	|	p		primitives
	|	x		variable
	|	$()$		unit introduction
	|	**true**		true
	|	**false**		false
	|	k		integer
	|	l		heap location
	|	el		array element
	|	**Many** v		!-introduction
	|	**fun** $fc \to v$		frac. cap. abstraction
	|	$v[f]$		frac. cap. specialisation
	|	(v, v')		pair introduction
	|	**fun** $x : t \to e$	$\mathsf{bind}\ x\ \mathsf{in}\ e$	abstraction
	|	**fix** $(g, x : t, e : t')$	$\mathsf{bind}\ g \cup x\ \mathsf{in}\ e$	fixpoint
	|	(v)	S	parentheses

e	::=			expression
	|	p		primitives
	|	x		variable
	|	**let** $x = e$ **in** e'	$\mathsf{bind}\ x\ \mathsf{in}\ e'$	let binding
	|	$()$		unit introduction
	|	**let** $() = e$ **in** e'		unit elimination
	|	**true**		true
	|	**false**		false
	|	**if** e **then** e_1 **else** e_2		if

	k		integer
	l		heap location
	el		array element
	$\mathbf{Many}\ e$		!-introduction
	$\mathbf{let}\,\mathbf{Many}\ x = e\,\mathbf{in}\ e'$		!-elimination
	$\mathbf{fun}\ fc \to e$		frac. cap. abstraction
	$e[f]$		frac. cap. specialisation
	(e, e')		pair introduction
	$\mathbf{let}\,(a, b) = e\,\mathbf{in}\ e'$	bind $a \cup b$ in e'	pair elimination
	$\mathbf{fun}\ x : t \to e$	bind x in e	abstraction
	$e\ e'$		application
	$\mathbf{fix}\,(g, x : t, e : t')$	bind $g \cup x$ in e	fixpoint
	(e)	S	parentheses

erased_v, ev	::=		values
	p		primitives
	x		variable
	$()$		unit introduction
	$\mathbf{true}$		true
	$\mathbf{false}$		false
	k		integer
	l		heap location
	el		array element
	(ev, ev')		pair introduction
	$\mathbf{fun}\ x \to er$	bind x in er	abstraction
	$\mathbf{fix}\,(g, x, er)$	bind $g \cup x$ in er	fixpoint
	(ev)	S	parentheses

erased, er	::=		expression
	p		primitives
	x		variable
	$\mathbf{let}\ x = er\,\mathbf{in}\ er'$	bind x in er'	let binding
	$()$		unit introduction
	$\mathbf{let}\,() = er\,\mathbf{in}\ er'$		unit elimination
	$\mathbf{true}$		true
	$\mathbf{false}$		false
	$\mathbf{if}\ er\,\mathbf{then}\ er_1\,\mathbf{else}\ er_2$		if
	k		integer
	l		heap location
	el		array element
	(er, er')		pair introduction
	$\mathbf{let}\,(a, b) = er\,\mathbf{in}\ er'$	bind $a \cup b$ in er'	pair elimination
	$\mathbf{fun}\ x \to er$	bind x in er	abstraction
	$er\ er'$		application
	$\mathbf{fix}\,(g, x, er)$	bind $g \cup x$ in er	fixpoint
	(er)	S	parentheses

C	::=			evaluation contexts
	$\mid$	$\mathbf{let}\ x = [-]\ \mathbf{in}\ er$	bind x in er	let binding
	$\mid$	$\mathbf{let}\ () = [-]\ \mathbf{in}\ er$		unit elimination
	$\mid$	$\mathbf{if}\ [-]\ \mathbf{then}\ er_1\ \mathbf{else}\ er_2$		if
	$\mid$	$([-], er)$		pair introduction
	$\mid$	$(ev, [-])$		pair introduction
	$\mid$	$\mathbf{let}\ (a, b) = [-]\ \mathbf{in}\ er$	bind $a \cup b$ in er	pair elimination
	$\mid$	$[-]er$		application
	$\mid$	$ev[-]$		application
Θ	::=			fractional capability environment
	$\mid$	$.$		
	$\mid$	Θ, fc		
Γ	::=			linear types environment
	$\mid$	$.$		
	$\mid$	$\Gamma, x : t$		
	$\mid$	Γ, Γ'		
Δ	::=			intuitionistic types environment
	$\mid$	$.$		
	$\mid$	$\Delta, x : t$		
σ	::=			heap
	$\mid$	$\{\}$		empty heap
	$\mid$	$\sigma \uplus \{l \mapsto m\}$		location l points to matrix m

$\boxed{\Theta \vdash f\ \mathsf{Cap}}$ Valid fractional capabilities

$$\frac{fc \in \Theta}{\Theta \vdash fc\ \mathsf{Cap}}\ \text{WF\_Cap\_Var}$$

$$\frac{}{\Theta \vdash 1\ \mathsf{Cap}}\ \text{WF\_Cap\_Zero}$$

$$\frac{\Theta \vdash f\ \mathsf{Cap}}{\Theta \vdash \frac{1}{2} \cdot f\ \mathsf{Cap}}\ \text{WF\_Cap\_Succ}$$

$\boxed{\Theta \vdash t\ \mathsf{Type}}$ Valid types

$$\frac{}{\Theta \vdash \mathbf{unit}\ \mathsf{Type}}\ \text{WF\_Type\_Unit}$$

$$\frac{}{\Theta \vdash \mathbf{bool}\ \mathsf{Type}}\ \text{WF\_Type\_Bool}$$

$$\frac{}{\Theta \vdash \mathbf{int}\ \mathsf{Type}}\ \text{WF\_Type\_Int}$$

$$\frac{}{\Theta \vdash \mathbf{elt}\ \mathsf{Type}}\ \text{WF\_Type\_Elt}$$

$$\frac{\Theta \vdash f\ \mathsf{Cap}}{\Theta \vdash f\ \mathbf{arr}\ \mathsf{Type}}\ \text{WF\_Type\_Array}$$

$$\frac{\Theta \vdash t\ \mathsf{Type}}{\Theta \vdash !t\ \mathsf{Type}}\ \text{WF\_Type\_Bang}$$

$$\frac{\Theta, fc \vdash t\ \mathsf{Type}}{\Theta \vdash \forall fc.t\ \mathsf{Type}}\ \textsc{WF_Type_Gen}$$

$$\frac{\Theta \vdash t\ \mathsf{Type} \quad \Theta \vdash t'\ \mathsf{Type}}{\Theta \vdash t \otimes t'\ \mathsf{Type}}\ \textsc{WF_Type_Pair}$$

$$\frac{\Theta \vdash t\ \mathsf{Type} \quad \Theta \vdash t'\ \mathsf{Type}}{\Theta \vdash t \multimap t'\ \mathsf{Type}}\ \textsc{WF_Type_Lolly}$$

$\boxed{\Theta; \Delta; \Gamma \vdash e : t}$ Typing rules for expressions

$$\frac{}{\Theta; \Delta; \cdot, x : t \vdash x : t}\ \textsc{Ty_Var_Lin}$$

$$\frac{x : t \in \Delta}{\Theta; \Delta; \cdot \vdash x : t}\ \textsc{Ty_Var}$$

$$\frac{\Theta; \Delta; \Gamma \vdash e : t \quad \Theta; \Delta; \Gamma', x : t \vdash e' : t'}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, x = e\, \mathbf{in}\, e' : t'}\ \textsc{Ty_Let}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash () : \mathbf{unit}}\ \textsc{Ty_Unit_Intro}$$

$$\frac{\Theta; \Delta; \cdot \vdash e : \mathbf{unit} \quad \Theta; \Delta; \Gamma \vdash e' : t}{\Theta; \Delta; \Gamma \vdash \mathbf{let}\, () = e\, \mathbf{in}\, e' : t}\ \textsc{Ty_Unit_Elim}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash \mathbf{true} : !\mathbf{bool}}\ \textsc{Ty_Bool_True}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash \mathbf{false} : !\mathbf{bool}}\ \textsc{Ty_Bool_False}$$

$$\frac{\Theta; \Delta; \Gamma \vdash e : \mathbf{bool} \quad \Theta; \Delta; \Gamma' \vdash e_1 : t' \quad \Theta; \Delta; \Gamma' \vdash e_2 : t'}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{if}\, e\, \mathbf{then}\, e_1\, \mathbf{else}\, e_2 : t}\ \textsc{Ty_Bool_Elim}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash k : !\mathbf{int}}\ \textsc{Ty_Int_Intro}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash el : !\mathbf{elt}}\ \textsc{Ty_Elt_Intro}$$

$$\frac{\Theta; \Delta; \cdot \vdash v : t \quad v \neq l}{\Theta; \Delta; \cdot \vdash \mathbf{Many}\, v : !t}\ \textsc{Ty_Bang_Intro}$$

$$\frac{\Theta; \Delta; \Gamma \vdash e : !t \quad \Theta; \Delta, x : t; \Gamma' \vdash e' : t'}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, \mathbf{Many}\, x = e\, \mathbf{in}\, e' : t'}\ \textsc{Ty_Bang_Elim}$$

$$\frac{\Theta; \Delta; \Gamma \vdash e : t \quad \Theta; \Delta; \Gamma' \vdash e' : t'}{\Theta; \Delta; \Gamma, \Gamma' \vdash (e, e') : t \otimes t'}\ \textsc{Ty_Pair_Intro}$$

$$\frac{\Theta; \Delta; \Gamma \vdash e_{12} : t_1 \otimes t_2 \quad \Theta; \Delta; \Gamma', a : t_1, b : t_2 \vdash e : t}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, (a, b) = e_{12}\, \mathbf{in}\, e : t}\ \textsc{Ty_Pair_Elim}$$

$$\frac{\Theta \vdash t' \,\mathsf{Type} \quad \Theta; \Delta; \Gamma, x : t' \vdash e : t}{\Theta; \Delta; \Gamma \vdash \mathbf{fun}\, x : t' \to e : t' \multimap t} \quad \textsc{Ty\_Lambda}$$

$$\frac{\Theta; \Delta; \Gamma \vdash e : t' \multimap t \quad \Theta; \Delta; \Gamma' \vdash e' : t'}{\Theta; \Delta; \Gamma, \Gamma' \vdash e\ e' : t} \quad \textsc{Ty\_App}$$

$$\frac{\Theta, fc; \Delta; \Gamma \vdash e : t}{\Theta; \Delta; \Gamma \vdash \mathbf{fun}\, fc \to e : \forall\, fc.t} \quad \textsc{Ty\_Gen}$$

$$\frac{\Theta \vdash f \,\mathsf{Cap} \quad \Theta; \Delta; \Gamma \vdash e : \forall\, fc.t}{\Theta; \Delta; \Gamma \vdash e[f] : t[f/fc]} \quad \textsc{Ty\_Spc}$$

$$\frac{\Theta; \Delta, g : t \multimap t'; \cdot, x : t \vdash e : t'}{\Theta; \Delta; \cdot \vdash \mathbf{fix}\,(g, x : t, e : t') : !(t \multimap t')} \quad \textsc{Ty\_Fix}$$

$\boxed{\langle \sigma, er \rangle \to \langle \sigma', er' \rangle}$ operational semantics

$$\frac{}{\langle \sigma, \mathbf{let}\,() = ()\,\mathbf{in}\, er \rangle \to \langle \sigma, er \rangle} \quad \textsc{Op\_Let\_Unit}$$

$$\frac{}{\langle \sigma, \mathbf{let}\, x = ev\,\mathbf{in}\, er \rangle \to \langle \sigma, er[x/ev] \rangle} \quad \textsc{Op\_Let\_Var}$$

$$\frac{}{\langle \sigma, \mathbf{if\,true\,then}\, er_1\, \mathbf{else}\, er_2 \rangle \to \langle \sigma, er_1 \rangle} \quad \textsc{Op\_If\_True}$$

$$\frac{}{\langle \sigma, \mathbf{if\,false\,then}\, er_1\, \mathbf{else}\, er_2 \rangle \to \langle \sigma, er_2 \rangle} \quad \textsc{Op\_If\_False}$$

$$\frac{}{\langle \sigma, \mathbf{fix}\,(g, x, er)\ ev \rangle \to \langle \sigma, er[x/ev][g/\mathbf{fix}\,(g, x, er)] \rangle} \quad \textsc{Op\_Let\_Fix}$$

$$\frac{}{\langle \sigma, (\mathbf{fun}\, x \to er)\ ev \rangle \to \langle \sigma, er[x/ev] \rangle} \quad \textsc{Op\_App}$$

$$\frac{er \to er'}{\langle \sigma, C[er] \rangle \to \langle \sigma, C[er'] \rangle} \quad \textsc{Op\_Context}$$

$$\frac{0 \le k_1, k_2}{\langle \sigma, \mathbf{matrix}\, k_1\, k_2 \rangle \to \langle \sigma \uplus \{l \mapsto M_{k_1,k_2}\}, l \rangle} \quad \textsc{Op\_Matrix}$$

$$\frac{}{\langle \sigma \uplus \{l \mapsto M_{k_1,k_2}\}, \mathbf{free}\, l \rangle \to \langle \sigma, () \rangle} \quad \textsc{Op\_Free}$$

$$\frac{}{\langle \sigma, \mathbf{share}\, l \rangle \to \langle \sigma, (l, l) \rangle} \quad \textsc{Op\_Share}$$

$$\frac{}{\langle \sigma, \mathbf{unshare}\, l\, l \rangle \to \langle \sigma, l \rangle} \quad \textsc{Op\_Unshare}$$

$$\frac{\sigma(l_1) = M_{k_1,k_2} \quad \sigma(l_2) = M_{k_2,k_3}}{\langle \sigma \uplus \{l_3 \mapsto M_{k_1,k_3}\}, \mathbf{gemm}\, l_1\, l_2\, l_3 \rangle \to \langle \sigma \uplus \{l_3 \mapsto M_{k_1,k_2}\, M_{k_2,k_3} + M_{k_1,k_3}\}, l_3 \rangle} \quad \textsc{Op\_Gemm}$$